been decided, the method may then search the XHTML code to find the anchor node for each instance of the XHTML structure. Finally, a generalized XPath expression is computed which matches all of them.

Once the general XPath expression has been computed, it is used to call the template for creating the group or atomic. In XSL, if we call a template using an XPath expression that matches more than one node, the template code gets run a number of times equal to the number of nodes in the nodeset described by the XPath expression. Thus, if we call the template using an XPath expression that matches 3 nodes, the template gets called 3 times, and a different node in the nodeset is used as the context node each time it is executed.

Therefore, once the generalizer has computed a general XPath expression, it can call a template for the creation of a certain type of group or atomic equal to the number of times a certain XHTML node structure appears in an XHTML page. Thus, in essence, the generalizer produces an XSL Template which creates a certain RML node and it gets called a number of times equal to however many instances of the corresponding XHTML structure occur.

When the generalization method is used, the hierarchical structure of the ARML is slightly different from the RML. The ARML and the RML will differ in structure in the case of generalized regions. This is because the ARML only contains a handful of examples of restructured XHTML, while the RML should contain all of them for a given page. For generalization, it is impossible to have the two match for the simple fact that the ARML is not dynamically created, while the RML is. If the number of groups or atomics changes in the RML

due to dynamism in the XHTML, this cannot be identically reflected in the ARML, which is really just a static set of instructions for the creation of a stylesheet.

In addition, the mapping given by the ARML in the case of generalization is incomplete. The goal of the generalizer is to compute the correct mapping and place it in the XSL, given several examples of sections in the XHTML that need to be mapped into a particular RML substructure specified by the user. However, when the information used to compute the mapping is still in its ARML form, the item-by-item correspondence between an RML node and its XHTML is not present. Now, the generalizer method in accordance with the invention will be described.

Figure 5 illustrates an embodiment of a generalizer method 110 in accordance with the invention. In step 112, the method may determine if the current node being processed has any "generalized" children. If it does have generalized children, then the method goes to step 114 in which the next generalized child is retrieved and the method recurses on the child to find other children or grandchildren (nested) that are generalized. As mentioned earlier, the generalization algorithm allows for nested generalization. It is a simple recursion, which processes generalization nodes in a bottom-up fashion. This reduces the problem of generalization to a two-case problem, where the generalization algorithm is dealing with either 1) paths which have not yet been generalized or 2) paths which have already been generalized. This avoids the problem of trying to generalize structures, which contain generalized nodes within them.

If the node does not have any generalized children, then the method determines if this is a case of anchor node sibling or not in step 116. This may be detected in the XPathPreProcessor

class. In particular, each ARML node has an "xhtmlpath" computed for it, which is the reference point from which all paths inside the node are defined. If the set of nodes to be generalized all end up with empty xhtmlpaths after pre-processing, then it is a case of anchor node sibling. This is because the xhtmlpath of the generalized node becomes relativized to be the common parent of all children of the example nodes, which means this cannot be used as an anchor.

If there is an anchor node sibling, then the method computes the anchors in step 118. Otherwise, the next step involves actually generalizing (combining) the paths in step 120. It is a requirement that the structure should be identical in all examples so there will be the same number of paths in each and they will occur in the same locations. These paths are matched up into sets across the examples so that all paths that occur in the same location in the examples are grouped/combined together. This set of paths is sent to a path-combining method, that is described below with reference to Figure 6, that computes a generalized XPath expression that matches all of them.

The next step in the generalization method after the paths have been combined is to re-anchor and untangle the replacement node if necessary. Thus, in step 122, the method determines if the anchor node is a sibling. If the anchor node is sibling, the reanchoring and untangling of that node if needed is carried out. Re-anchoring is a simple matter in that the paths need to be made relative to a different node by using following-sibling and previous-sibling axes in step 124. However, if there is a case of anchor-node-parent inside of a case of anchor-node-sibling during a nested generalization, the interior nodes will have an anchor node higher in the tree than the exterior nodes' anchor nodes. In that case, special handling is required to re-anchor